Amendments to the Drawings

The attached sheet of drawings includes changes to Fig. 17. This sheet replaces the original sheet of Fig. 17. In Fig. 17, extractors 640 have been amended to extractors 650. No new matter is added.

Attachment: Replacement sheet for Fig. 17

REMARKS

The specification, drawings, and claims 1 and 17 are amended herein. Claims 1-20 remain pending in the captioned case. Further examination and reconsideration of the presently claimed application are respectfully requested.

Objection to the Drawings

An objection was lodged against the drawings for informalities. Specifically, the reference character "640" was used to define both CPU count and extractors in Fig. 17 and corresponding text of the specification. In response thereto, the specification and Fig. 17 are amended to note the extractors as reference character "650." In addition, reference character "222" in Fig. 2B was not defined in the specification. In response thereto, the specification has been amended to identify the prepared map object 222. Accordingly, removal of these objections are respectfully requested.

Objection to Specification

An objection was lodged against the specification for containing an embedded hyperlink or other browser-executable code. Applicants respectfully disagree. Upon review of the specification, Applicant can find no embedded hyperlink nor other browser-executable code. The website identified on page 6, lines 14-15 of the specification is not a hyperlink. Accordingly, removal of this objection is respectfully requested.

An additional objection was lodged against the specification for the use of undefined acronyms. In response thereto, the specification has been amended to properly define the acronyms at their first use in the specification. Accordingly, removal of this objection is respectfully requested.

Section 101 Rejection

Claims 17-20 were rejected under 35 U.S.C. § 101 as directed to non-statutory subject matter. In response thereto, claim 17 is amended in a manner believed to obviate this rejection. Accordingly, removal of this rejection is respectfully requested.

Section 102 Rejection

Claims 1-9 and 17-20 were rejected under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,999,729 to Tabloski, Jr. et al. (hereinafter "Tabloski"). The standard for "anticipation" is one of fairly strict identity. A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art of reference. *Verdegaal Bros. v. Union Oil Co. of California*, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987); MPEP 2131. Furthermore, anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, as arranged in the claim. *W.L. Gore & Assocs. V. Garlock*, 721 F.2d 1540, 220 USPQ 303 (Fed. Cir. 1983). Using these standards, Applicants submit Tabloski fails to disclose each and every element of the currently pending claims, some distinctive features of which are set forth in more detail below.

While both the present invention and Tabloski use dataflow graphs to describe a parallel application, Tabloski describes a different type of system. Dataflow graphs are commonly used to describe systems based on the pipes and filters architectural style. Tabloski uses a highlevel language (C++) to compile a plurality of graphical map components organized and linked by the user.

That is, Tabloski uses the dataflow graphs to generate code in a high level language (C++) using the organization and links designated by the user. This high level code is then compiled, using a compiler for the high level language, into an executable program.

The present invention uses an XML-based representation dataflow graph. This XML describes the dataflow operators (components called map components and externally defined sub graphs called map processes), their configurations and how they are lined to each other. The present invention does have a compiler; however, this compiler does not translate code into executable instructions. It serves to construct an internal representation of the graph where composite operators are in-lined and the ports are configured with the appropriate schema information. This internal representation is used to validate and optimize the graph before execution. At execution time, this internal representation of the graph is used to load and configure the map components, link the ports, and to start each map component executing on a separate thread as part of a single process.

Additionally, in one aspect the present invention can support user-defined operators. New map components can be added to the component library using the host language (Java) or by composing map components into map processes using the XML dataflow graph representation.

The execution environments for Tabloski and the present invention are also different. While Tabloski and the present invention are designed to execute applications in parallel on systems with multiple processors. Applications in the present invention are targeted at SMP systems and are executed on threads in a single process image. Data is transferred between operators using connector objects (queues). Operators enqueue and dequeue data using input and output ports. A separate thread is used to monitor the engine for deadlocks.

The Tabloski invention achieves parallelism by executing objects concurrently as separate processes. These processes can execute on different nodes of a networked system. This type of system requires a completely different way for managing the execution of the different processes (their execution control object) and for passing data between processes.

Turning to the specific invention referenced in claim 1, while Tabloski does relate to a method for executing a dataflow application, if does so in a different manner as noted above. The Tabloski invention does not seem to support the use of composite operators or composite

components. The invention of claim 1 calls for the use of composite components. This approach provides an extension mechanism and allows for the reuse of dataflow graphs.

The Tabloski dataflow application does not use threads to execute the objects in the dataflow. Further, the Tabloski dataflow application uses a highlevel language (C++) to compile a plurality of graphical map components organized and linked by the user. As such, Tabloski does not execute map components on a separate thread.

Turning to the dependent claims, Tabloski does not suggest sythesizing data type information or supporting polymorphic ports (Claim 4). Tabloski does not disclose the use of hierarchical ports (Claim 5). Tabloski supports vertical parallelism where a port has individually addressable subports. A key point with the invention of claim 5 is the ability to support very fine grained parallelism. Many dataflow systems, such as Tabloski, move streams of records between operators. The invention of claim 5 supports moving both records and discrete fields.

Nowhere does Tabloski disclose the use of token batches (Claim 7). It does not appear that Tabloski discloses the use of multi-state null value tokens (claims 8, 9, 17 -20).

For at least the reasons set forth above, claims 1-9 and 17-20 are believed patentably distinct over the cited art. Accordingly, removal of this rejection is respectfully requested.

Section 103 Rejection

Claims 10-16 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Tabloski in view of U.S. Patent Application Publication No. 2004/0025164 to Ma et al. (hereinafter "Ma"). While Tabloski and Ma do provide for deadlock detection, they are substantially different from the inventions described in claims 10 -16. Tabloski describes a different type of deadlock detection mechanism for managing deadlock across multiple processes. Tabloski creates an execution control object on each node of the networked processing environment to communicate with the execution objects. Claim 10 does not use a control object. Tabloski does not monitor deadlock on a thread level. The execution objects communicate blockage to execution control object. In the present

deadlock monitoring invention, the monitoring thread watches the data queues and does not require communication from the operators. It expands the queues if necessary based on information it has collected in the wait graph. In Ma's invention there is no notion of the relationships between the threads that are sharing a mutually exclusive resource. Ma uses queues and not wait graphs. Additionally, Ma provides no remedy for the deadlock situation it detects. Ma describes a method for detecting deadlocks among threads where the threads are all waiting on a mutually exclusive resource.

A key advantage to the use of deadlock detection in the present invention is to provide an automatic mechanism for dealing with different rates of flow of the tokens through the dataflow graph. Some operators need more time to process data than others or may need to aggregate data, so that they consume data at a different rate than they emit it. The deadlock detection algorithm is all about balancing the queue sizes for the connectors (links) between the operators. Claim 10 bears out these distinctions.

Turning to the dependent claims, Ma does not disclose the use of wait graphs (e.g. claim 13). Additionally, Ma monitors the threads using a shared resource. Ma is concerned with resource contention. Claim 13 is concerned with blockage data flowing through the queues. In claim 14, Tabloski does not use queue objects (links or connectors) between execution objects. Because the Tabloski system is based on multiple objects over a network of processors they must manage blockages in the data flow differently.

The queues of the present invention allow for multiple downstream consumers on the same stream of data. Tabloski does not use queues, so outbound ports in Tabloski buffer data. These buffers can be expanded in some cases to deal with blockage. In the present invention the ports build batches of tokens and enqueue them on the outbound queue. The queues can be expanded if a data producer is producing data at a rate greater than one of the downstream consumers.

For at least the reasons set forth above, claims 10-16 are believed patentably distinct over the cited art. Accordingly, removal of this rejection is respectfully requested.

CONCLUSION

The present amendment and response is believed to be a complete response to the issues raised in the Office Action mailed December 31, 2007. In view of the amendments and remarks herein, Applicants assert that pending claims 1-20 are in condition for allowance. If the Examiner has any questions, comments or suggestions, the undersigned attorney earnestly requests a telephone conference.

No fees are required for filing this amendment; however, the Commissioner is authorized to charge any additional fees which may be required, or credit any overpayment, to Daffer McDaniel, LLP Deposit Account No. 50-3268.

Respectfully submitted,

/Charles D. Huston/
Charles D. Huston
Reg. No. 31,027
Attorney for Applicant(s)

Customer No. 35617
Date: June 2, 2008